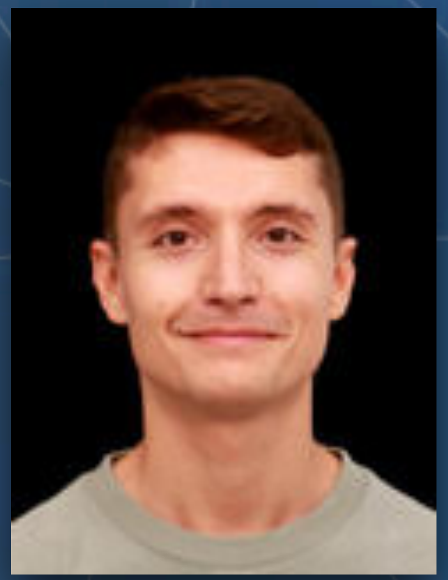
NEURAL INFORMATION PROCESSING SYSTEMS

μPC: Scaling Predictive Coding to 100+ Layer Networks


Francesco Innocenti


El Mehdi Achour


Christopher L. Buckley

UNIVERSITY OF OXFORD

sussex NEUROSCIENCE

UNIVERSITY OF SUSSEX

UM6P

College of Computing

NeurIPS 2025

TL;DR

We introduce μPC, a reparameterisation of PC networks that enables the stable training of 100+ layer ResNets on simple tasks with zero-shot hyperparameter transfer

Overview

1. Introduction
2. Problems with standard PC
3. μPC and experiments
4. Future directions
5. μPC code

Overview

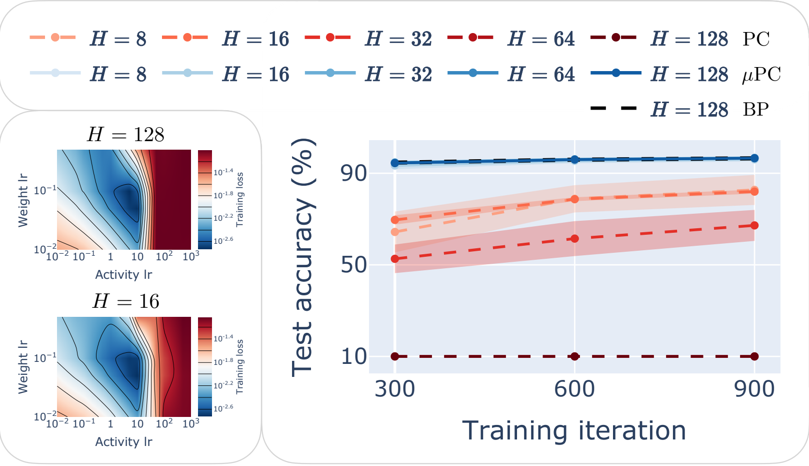
1. **Introduction**
2. Problems with standard PC
3. μPC and experiments
4. Future directions
5. μPC code

Motivation

- Backpropagation (BP) is known to be energy inefficient and biologically implausible
- This motivated many brain-inspired learning algorithms such as predictive coding (PC), equilibrium propagation, and forward learning, among many others
- The challenge has been to scale these algorithms to deep models and large datasets[1]
- We focus on PC, studying why it is hard to scale
- We propose a new reparameterisation of PC networks (PCNs) that, for the first time, allows stable training of 100+ layer networks on simple tasks with zero-shot hyperparam. transfer

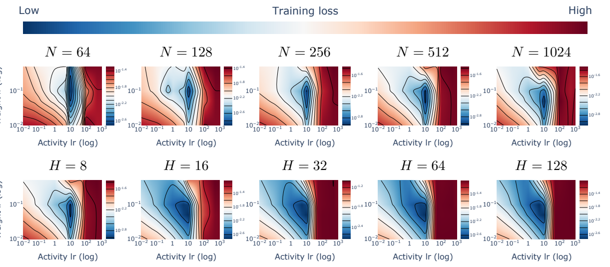
[1][Pinchetti et al. '25, ICLR]

Main contributions

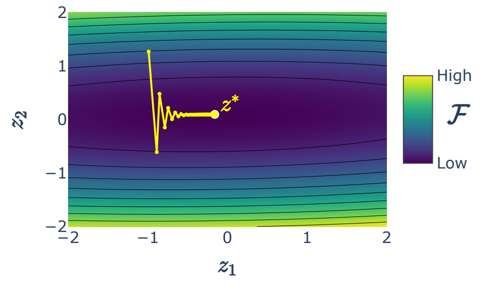


- We show that μPC, which reparameterises PCNs using "Depth-μP", allows stable training of very deep (100+ layer) ResNets on simple classification tasks with competitive performance and little tuning compared to current benchmarks

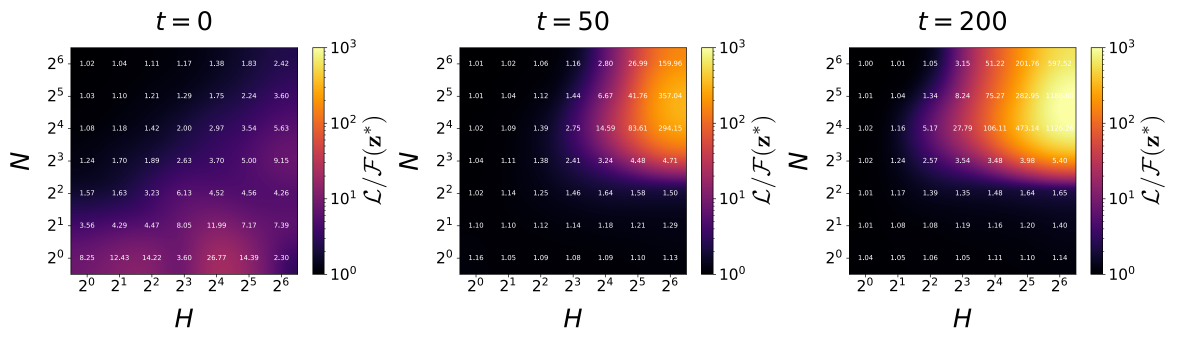
- μPC also empirically enables zero-shot transfer of both the weight and activity learning rates across model widths and depths



- We achieve these results by a theoretical and empirical analysis of the inference landscape and dynamics of PCNs



- To better understand μPC, we study a theoretical regime where it effectively implements BP; yet, we find that μPC can successfully train deep networks far from this regime

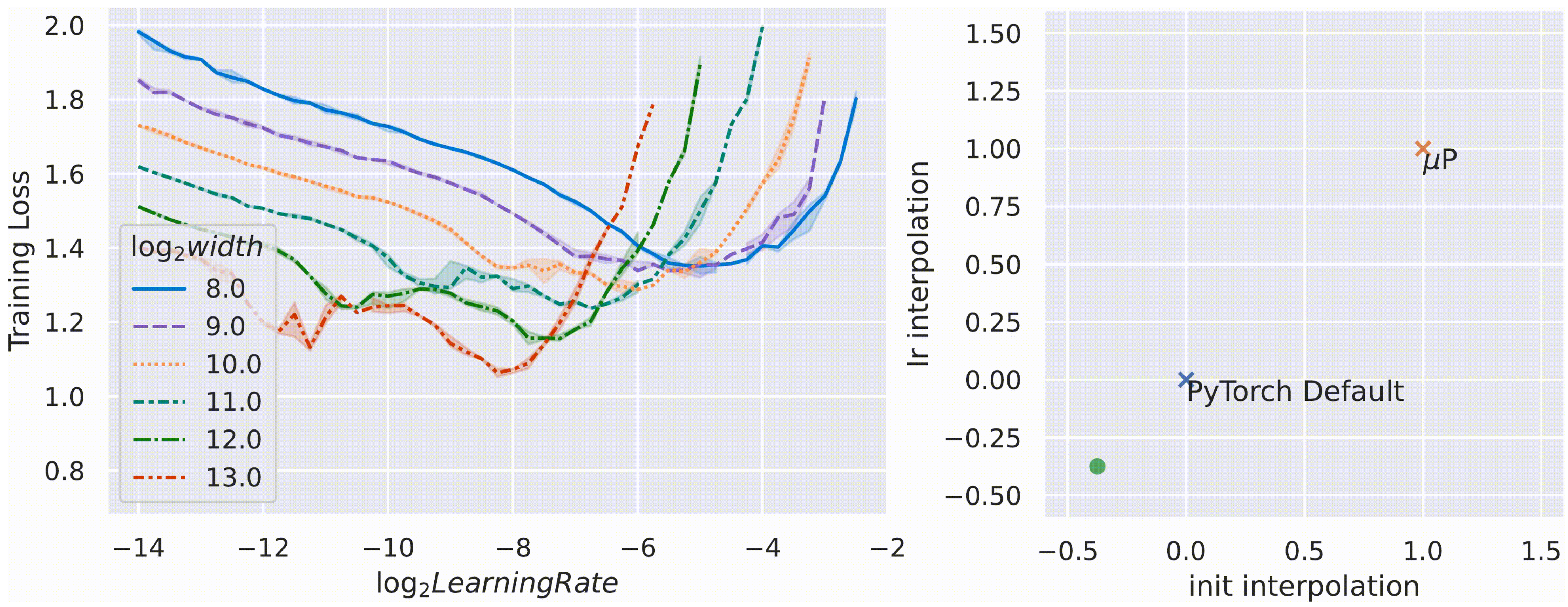


Background: μP

- The maximal update parameterisation (μP) is a theoretical prescription for how to scale models such that the order of activation or feature updates at every layer remains stable (“order 1”) with the model size (e.g. width and depth) [Yang & Hu ’21, ICML]

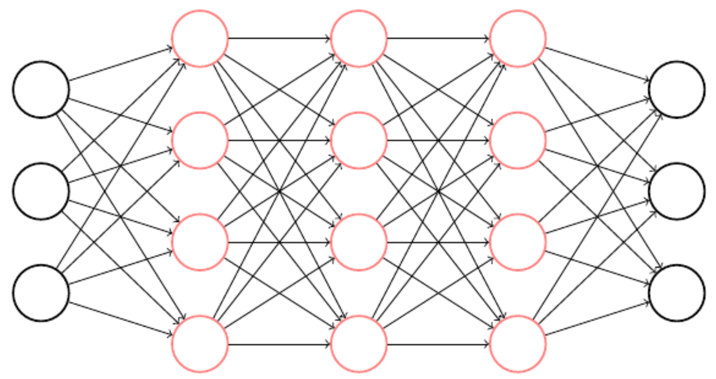
- μP was originally developed for model width and more recently extended to depth (“Depth-μP”) [Yang et al. ’24, ICLR; Bordelon et al. ’24, ICLR; Dey et al. ’25, NeurIPS]

- It is not only the training dynamics that remain stable across model sizes but also hyperparameters such as learning rate, thus enabling zero-shot transfer from small to large models



[Yang et al. ’21, NeurIPS]

Background: PC

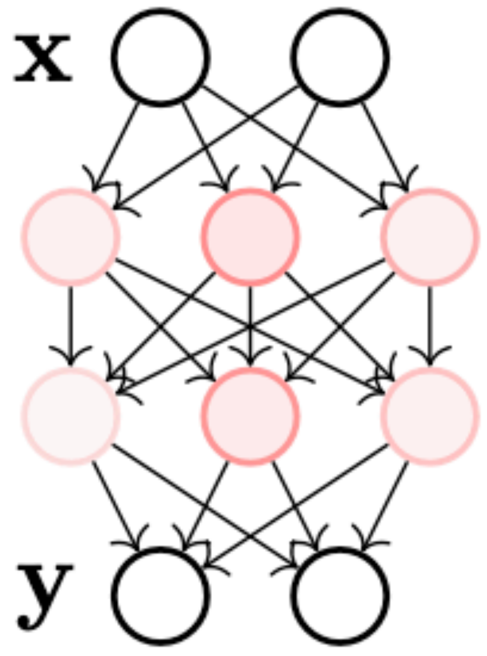


- PC networks minimise an energy function which is a sum of layer-wise prediction errors [Millidge et al. '21, arXiv], such as the following:

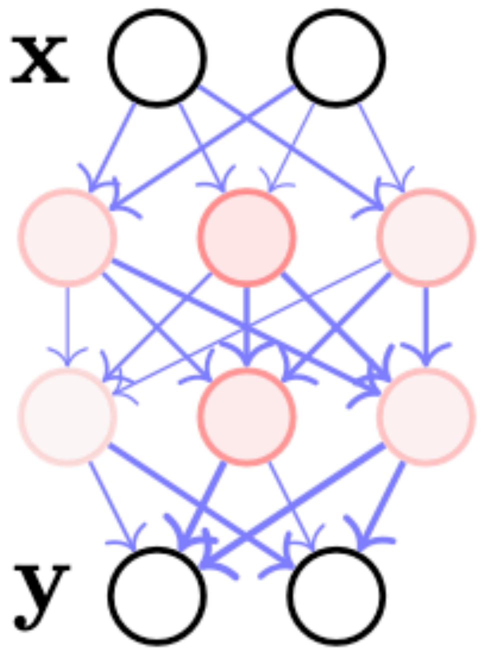
$$\mathcal{F} = \sum_{\ell=1}^{L} \frac{1}{2} ||\mathbf{z}_\ell - a_\ell \mathbf{W}_\ell \phi_\ell(\mathbf{z}_{\ell-1}) - \tau_\ell \mathbf{z}_{\ell-1}||^2$$

- The standard parameterisation uses unit premultipliers and initialises weights with variance proportional to 1/input_dim

- To train PCNs, we minimise the energy in two alternating phases: first with respect to the activities (**inference**) and then with respect to the weights (**learning**)



Infer: $\mathbf{z}^* = \arg\min_{\mathbf{z}} \mathcal{F}(\boldsymbol{\theta}_t, \mathbf{z})$



Learn: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \mathbf{P}_t \nabla_\theta \mathcal{F}(\boldsymbol{\theta}_t, \mathbf{z}^*)$

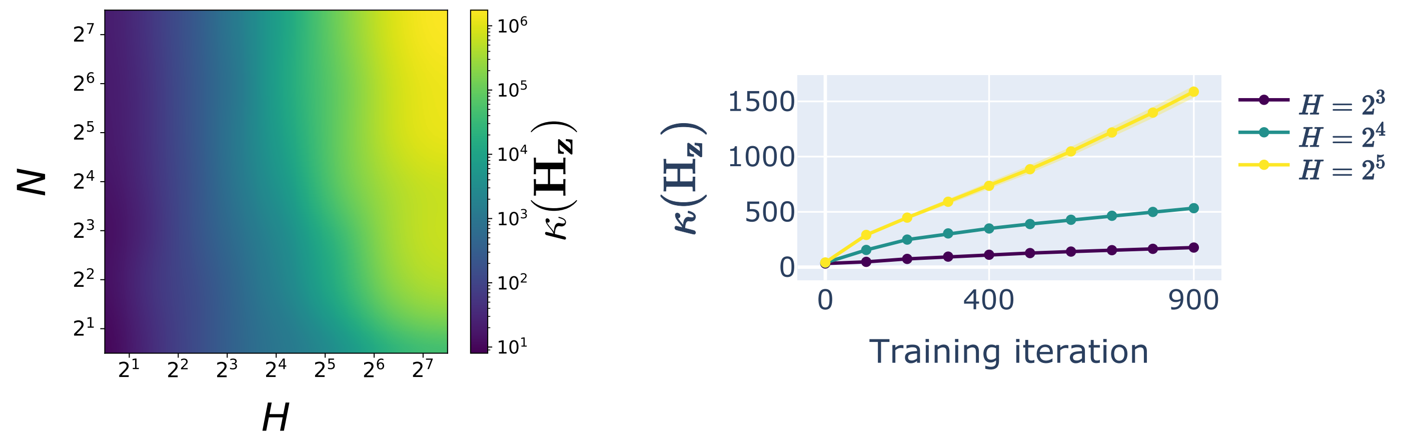
Overview

1. Introduction
2. **Problems with standard PC**
3. μPC and experiments
4. Future directions
5. μPC code

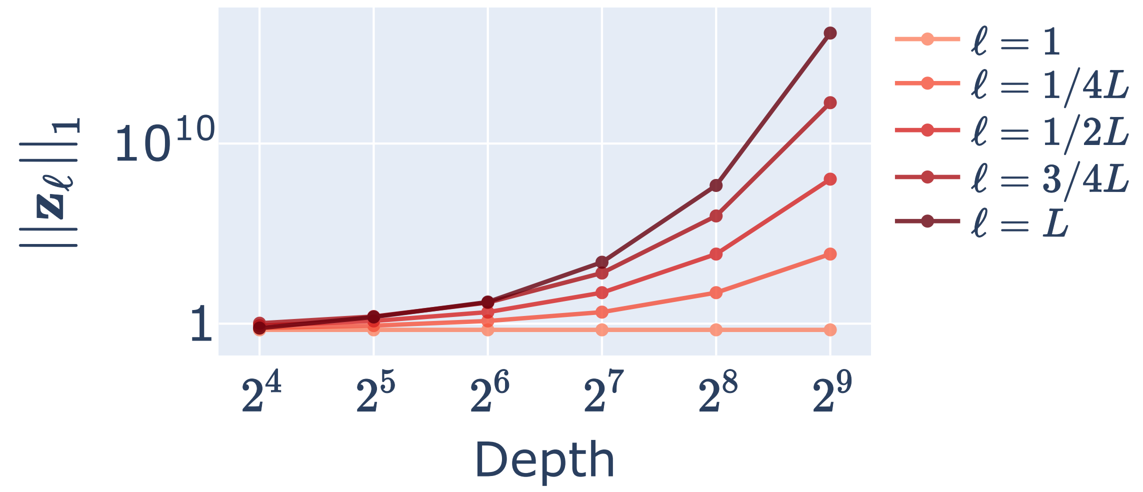
Problems with standard PC

- We show that PCNs become practically impossible to train at large scale due to a combination of 2 main factors:

1. The inference landscape grows increasingly ill-conditioned with the model size (particularly depth) as well as training time



2. The forward pass of standard PCNs tends to vanish/explode with depth (depending on the specific model)



- To cut a long story short, we find that there seems to be a tradeoff between these two pathologies - we prioritise forward pass stability as it appears more crucial for training

Overview

1. Introduction
2. Problems with standard PC
3. **μPC and experiments**
4. Future directions
5. μPC code

μPC

- To ensure forward pass stability with depth, we reparameterise the PC energy for ResNets (tau=1 for all hidden layers) with some of Depth-μP scalings ("**μPC**")

$$\mathcal{F} = \sum_{\ell=1}^{L} \frac{1}{2} ||\mathbf{z}_\ell - a_\ell \mathbf{W}_\ell \phi_\ell(\mathbf{z}_{\ell-1}) - \tau_\ell \mathbf{z}_{\ell-1}||^2$$

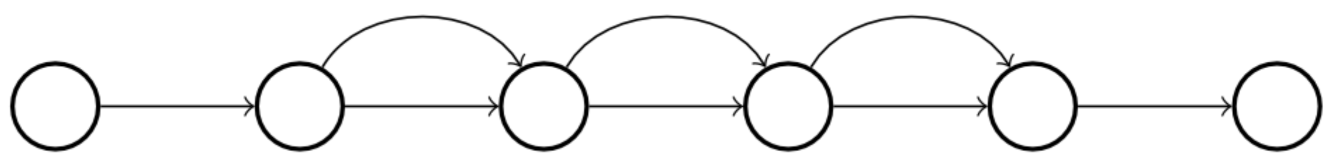


Table 1: **Summary of parameterisations.** Standard PC has unit layer premultipliers and weights initialised from a Gaussian with variance scaled by the input width at every layer $N_{\ell-1}$. μPC uses a standard Gaussian initialisation and adds width- and depth-dependent scalings at every layer.

	a_1 (input weights)	a_ℓ (hidden weights)	a_L (output weights)	b_ℓ (init. variance)
PC	1	1	1	$N_{\ell-1}^{-1}$
μPC	$N_0^{-1/2}$	$(N_{\ell-1}L)^{-1/2}$	N_{L-1}^{-1}	1

- We essentially rescale the residual branches by a depth-dependent factor

Experimental setup

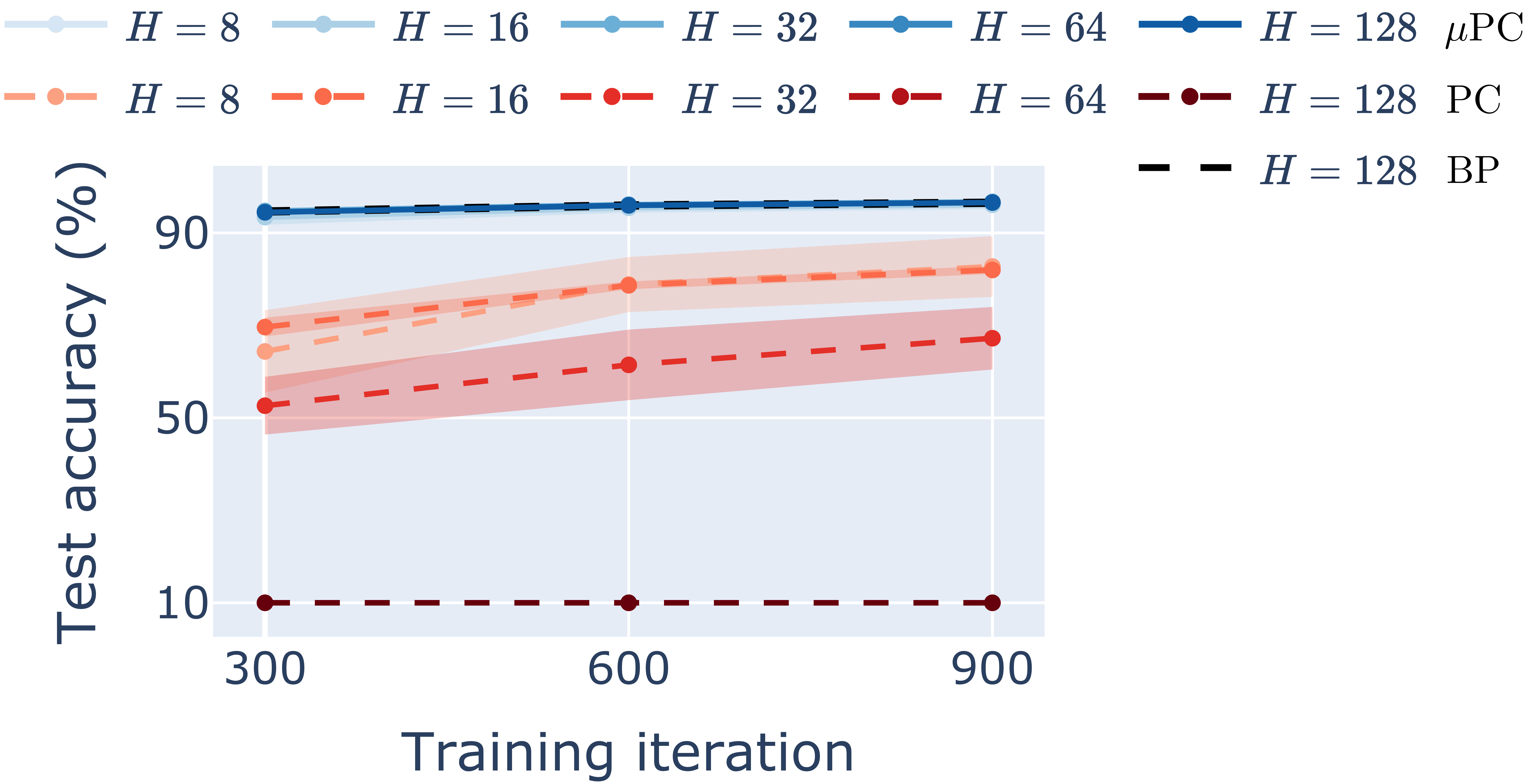
- We trained fully connected ResNets on simple classification tasks (MNIST, Fashion-MNIST & CIFAR-10) with the main goal of testing whether μPC could train deep PCNs

- All networks used as many inference steps as number of hidden layers

- In contrast to existing benchmarks [Pinchetti et al. ’25, ICLR], no optimisation techniques including momentum, nudging, weight decay, etc. were used

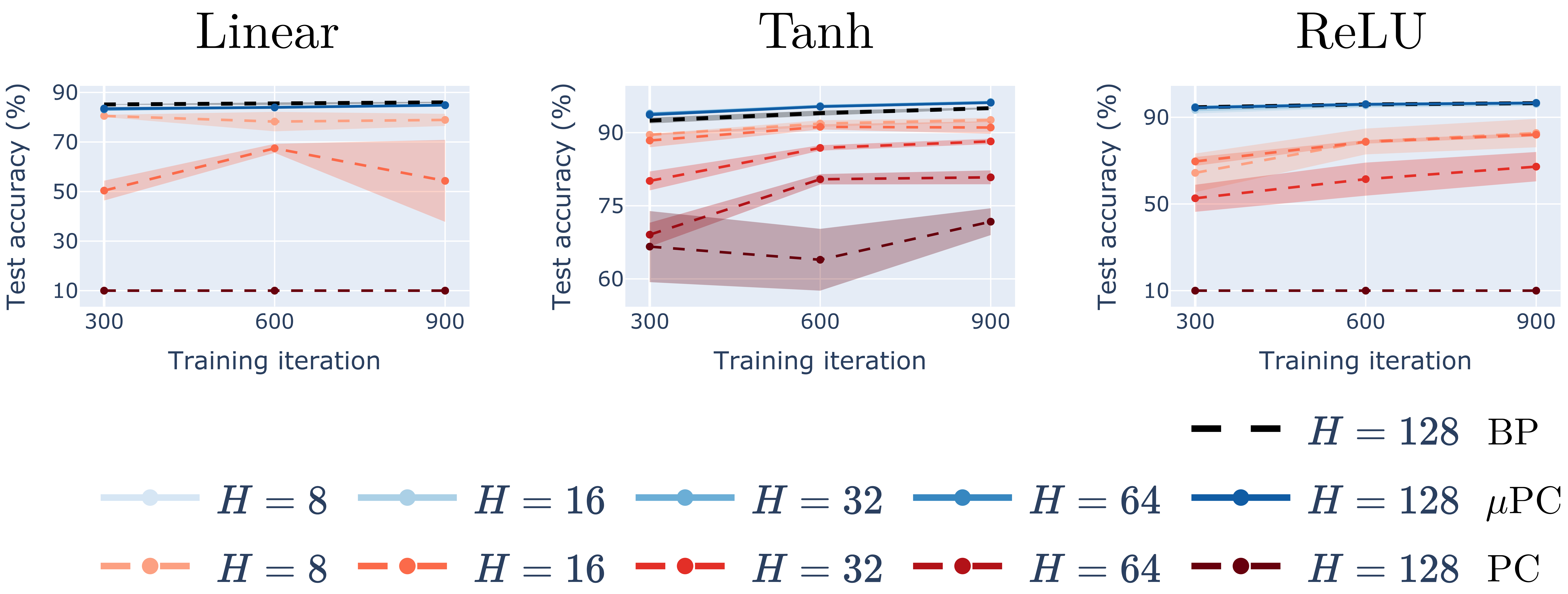
Experimental results

- We first trained ResNets of varying depth to classify MNIST for one epoch, finding that μPC allowed stable training of 100+ layer nets at the same performance as BP



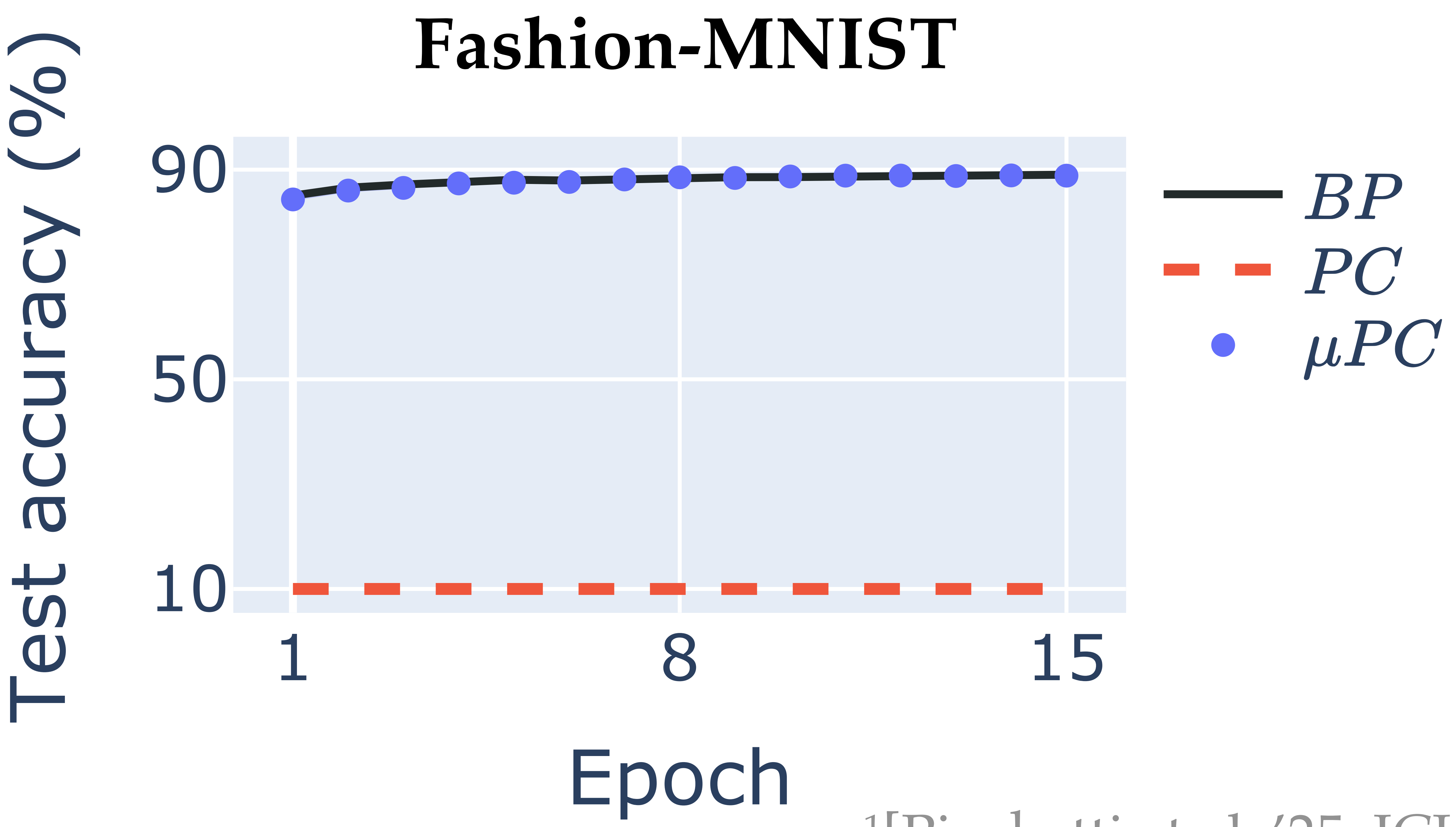
Experimental results

- Results were consistent across different activation functions



Experimental results (128 layers)

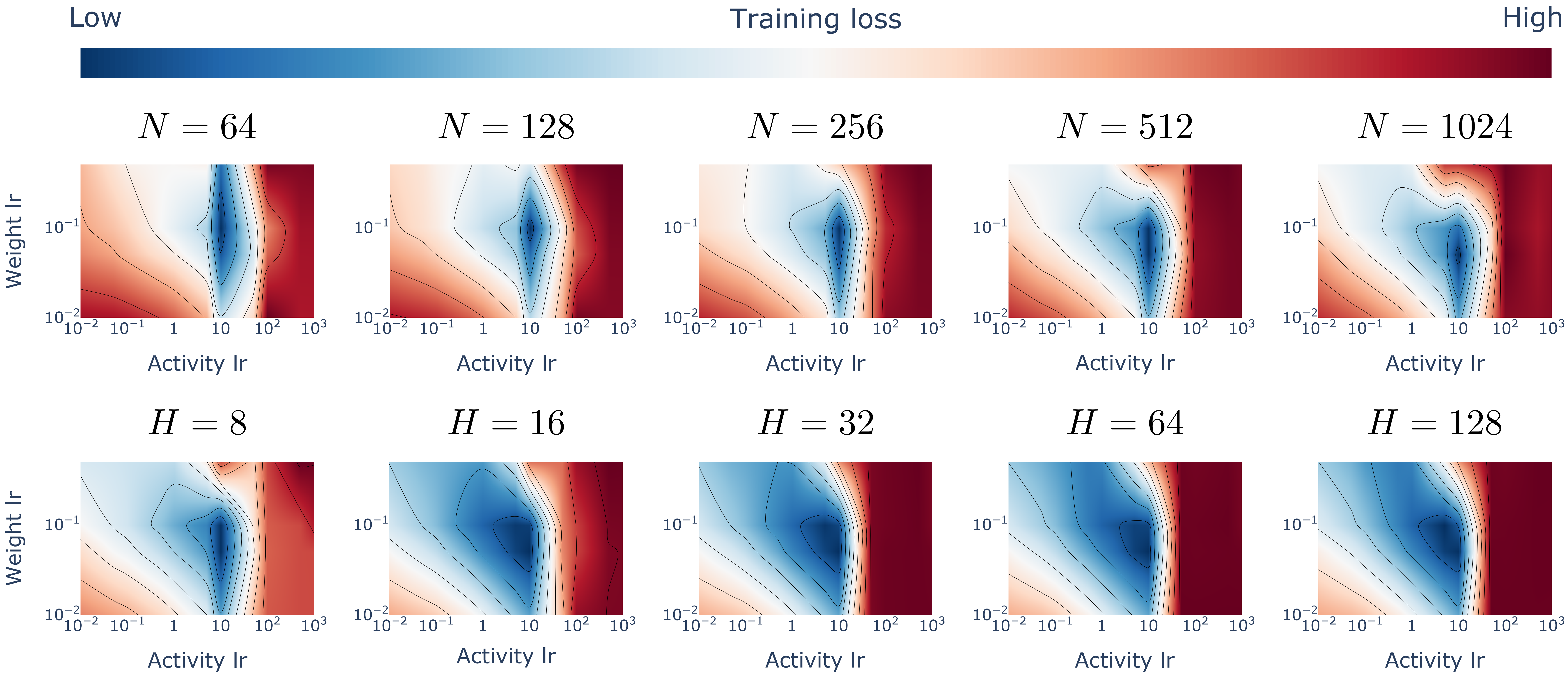
- ~98% on MNIST is achieved in 5 epochs (5x faster than current benchmark[1])
- ~89% on Fashion-MNIST is achieved in ~15 epochs (close 2x faster[1])
- All with little tuning (i.e. no advanced techniques and learning rate transfer)



[1][Pinchetti et al. '25, ICLR]

Experimental results

- Surprisingly, we also find that μPC enables zero-shot transfer of **both the weight and activity learning rates** across model widths N and depths H



Overview

1. Introduction
2. Problems with standard PC
3. μPC and experiments
4. **Future directions**
5. μPC code

Future directions

- Trying to extend μPC to more complicated architectures and datasets

- Understanding why μPC is successful despite the ill-conditioning of the inference landscape, and whether this might be necessary for more complicated datasets

- Investigating whether similar local algorithms such as equilibrium propagation would also benefit from similar reparamterisations

- For details and other results, check out the paper

Overview

1. Introduction
2. Problems with standard PC
3. μPC and experiments
4. Future directions
5. **μPC code**

Implementing μPC

https://thebuckleylab.github.io/jpc/examples/mupc/

JPC jpc

Search

thebuckleylab/jpc
v1.0.0 48 2

jpc
Getting started

How it works
Basic usage
Advanced usage

Examples
Introductory
Advanced
Hybrid PC
Linear theoretical energy
μPC
Hyperparameters
Dataset
Creating a μPC model
Train and test
Run

μPC

Open in Colab

This notebook demonstrates how to train residual networks with **μPC**, a reparameterisation of PC that allows stable training of very deep (100+ layer) networks while also enabling zero-shot hyperparameter transfer (see Innocenti et al., 2025).

```
%%capture
!pip install torch==2.3.1
!pip install torchvision==0.18.1
```

```
import jpc

import jax.random as jr
import equinox as eqx
import equinox.nn as nn
import optax

from jax.tree_util import tree_leaves
from typing import List
```

[Innocenti et al. '24, arXiv]

Thank you!

El Mehdi
Achour

Christopher
L. Buckley

UNIVERSITY OF OXFORD

sussex NEUROSCIENCE

UNIVERSITY OF SUSSEX

UM6P College of Computing